

A JavaScript Curriculum for High School Students to Explore Computer Science

Ching-Yu Huang and Mayra Bachrach

Abstract—Many students do not have the opportunity to experience computer science related courses in high school due to the lack of resources at their schools. Meanwhile, there are a large number of unfilled computing related jobs nationally. Universities and colleges in the United States are not producing enough Computer Science graduates to meet the demand for jobs—and, conversely, a number of students lured by the abundance of jobs in the field are entering undergraduate Computer programs with insufficient knowledge of the nature or rigor of the discipline, causing them to lose interest after the first year. Summer programs at universities and colleges offer students opportunities to learn about Computer Science, but many of the programs are meant to broaden the knowledge of students who are already studying Computer Science at their high schools by focusing on a specialized area in the discipline such as cybersecurity or game programming. These summer programs are costly and often beyond the reach of economically disadvantaged students. This article outlines a 5 day curriculum for a Computer Science summer program for high school students. Rather than focusing on deepening students' understanding of specialized topics, the curriculum is instead intended to expose students to foundational concepts in computer science using a hands-on, engaging approach. This foundational understanding of Computer Science will aide participating students in making informed choices about potential academic and career paths in Computer Science—and equip them with some baseline knowledge of the field should they choose to pursue a Computer Science degree.

Index Terms—Computer science, curriculum, exploring, summer program.

I. INTRODUCTION

Computers have become pervasive in our daily lives over the past decade, with even the very definition of a computer expanding to include mobile devices, smart home appliances, cloud technology, and more. Entire new industries are born and grow each year—and as a result, the demand for people to design and develop computing technology is growing rapidly. In the United States today, there are 500,000 open jobs in Computer Science—including over 21,000 in New Jersey, with an average salary of \$102,000 [1]. By 2020, an estimated one million computer programming jobs in the U.S. will go unfilled, according to the U.S. Bureau of Labor Statistics [2].

Currently, the curriculum in most U.S. high schools covers only basic sciences – biology, chemistry, physics and math. Some schools offer computer-related courses, the most common of which are “computer applications” and AP Computer Science. Many computer application courses focus

on how to use software such as MS Office, which won't help students to understand even the basics of computer science. While AP Computer Science, which is based on Java, is a better course to understand and experience programming, only 34% of NJ high schools offered AP Computer Science in 2016—and less than 1% of NJ high school students took the AP Computer Science A exam [3]. Clearly, not all students have the opportunity to learn Computer Science—or to pursue the valuable and rewarding careers that the field offers. Without experiencing or studying the main subject in a field such as computer science, students cannot be expected to select the field as a major in college or as a career. Moreover, even if students do pursue computer science, without earlier exposure they are less likely to thrive in their major. Many college students transfer out of STEM majors, especially Computer Science [4], because they don't experience computer science courses prior to entering the major and have unrealistic expectations when they enrolled in the program.

Many colleges and institutes offer summer programs or camps for high school students to learn specific subjects, such as programming, animation, photo effects, robotics, etc. These programs are expensive with price ranges from near \$1000 to \$2500 per week for one to three weeks. Some of the more selective programs even require students to submit PSAT, SAT, essay and recommendation letters similar to those required for college applications. Many disadvantaged families cannot afford for their children to attend these programs—and these students are less likely to have strong PSAT or SAT scores because their families cannot afford tutoring. These students' extracurricular activities are also often limited because they need to work in the summer, are unable to attend the residential programs, parents have to work and cannot drive them, or they have no transportation to attend programs outside their immediate area.

A multimedia curriculum was previously designed [5] and successfully implemented to expose middle school students to computer Science [6]. This paper expands and builds on this previous work by proposing a 5-day summer program that may be utilized by local universities to help high school students explore foundational concepts in computer science. Each day will consist of 5 hours of learning and hands-on lab exercises. The target students will be expected to have completed and passed Algebra II, but will not be expected to have any computer background.

II. CURRICULUM

The most important fundamental concept in computer science is programming. Therefore, most colleges list programming courses as the core courses required for

freshman to advance into higher level coursework. If a student is not interested in or skilled at basic programming, it is difficult for them to survive this first year and continue their studies in computer science. Even though AP Computer Science is offered in many high schools, without some initial exposure, it may be too much for some students to learn subjects such as Object Oriented programming, Inheritance, and Polymorphism. Some students might just want to experience basic coding and see if they are interested before they commit to take the year-long AP Computer Science course and the AP exam.

The summer course proposed here covers basic understanding of HTML and Java Script, as well as basic logic concepts such as how to identify and develop a solution to a problem. Students will learn how to write code in the correct syntax and how to fix simple coding errors (debugging skills). The summer program curriculum introduces the basic concepts and design assignments for students to do exercises within the 5-day program.

A. Students Learning Outcome (SLO)

After the summer program, students should be able to do the followings:

- Design basic web pages using HTML and JavaScript
- Define variable and explain the data types
- Describe and demonstrate problem-solving techniques
- Design, code, execute, test, debug, analyze, and document programs
- Write JavaScript programs with if-else and loop statements
- Write JavaScript programs using one-dimension array

B. Programming Feature Requirements

The basic program topics include variables and keywords, variable scope, assignment and computation, logic conditions, if-else statements, loops, arrays and indexes, methods (functions), and parameter passing.

A computer language requires a compiler to check the syntax and run the code to generate the results. For beginners, it is better to have a Graphic User Interface (GUI) tool that can make it easy for them to edit and visualize the code, indicate the locations of syntax and runtime errors, and see the results. They can learn how to change color in HTML using JavaScript [7].

Because these students will be beginners, it will be more interesting to them if they can see the results of their coding immediately. Hyper Text Markup Language (HTML) is the primary language for creating Web pages. It has several features that can interact with the user by taking input and generating the results immediately. However, HTML cannot do calculation and it doesn't have functions to work on complicated tasks, so it is not a real programming language. JavaScript is the programming language of HTML and the Web. In order to trigger beginners' interest in programming, both HTML and JavaScript are adopted for the summer program because it is easier for students to visualize their coding results on the browsers that most students are already familiar with.

To reduce the installation time and avoid confusion, the Online JavaScript Editor (OJSE) on Tutorialspoint.com [8] is used in this paper based on the factors mentioned previously.

Eclipse, a popular Integrated Development Environment (IDE) that supports different languages included JavaScript is not adopted because it requires installation which is dependent on each user's computer and Operating System. It might take instructors and students hours or days to fix problems before they can even start coding. As shown in Fig. 1, the OJSE allow users to edit their code and see the immediate results at the same time on the same web page. There is also a red indicator if there is a syntax error, and the error message will be displayed when the user moves the mouse over the indicator.

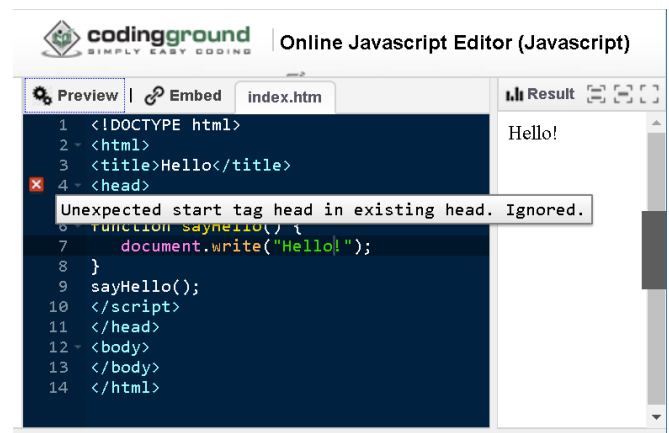


Fig. 1. Online JavaScript editor on tutorialspoint.com.

C. Programming Language

HTML uses tags <> for formatting the display on the browser, so it is not considered a regular computer language because it doesn't have functions. Therefore, only the basic tags will be covered. The HTML tags are not case sensitive.

- <HTML> and </HTML> are to begin and end the web page.
- <TITLE> and </TITLE> are to show the title of the web page.
- <HEAD> and </HEAD> are for the head elements.
- <SCRIPT> and </SCRIPT> are for JavaScript.
-
 is the tag to display at new line.
- <A HREF> and are to launch another web page.
- and are to change font size, color and style.
- <TABLE>, </TABLE>, <TR> and <TD> are to design tables with rows and data cells.
- <INPUT type='text'> is to take user's input typing on the browser
- <INPUT type='range'> has a graphic interface for user to set a number within a range.
- <FORM> and </FORM> are to define a form that is used to collect user input.
- <INPUT type='submit'> defines a submit button for submitting the form.

JavaScript is the main language proposed in this paper because its features meet the requirements we defined in section I. JavaScript is traditionally considered to be a client-side (front-end) interpreter language in which the source code examples can be run by the browser directly. However, it can be also considered a server-side (back-end) language under recent frameworks such as Node.js. In this paper, we will focus on only the client-side uses because

server-side applications are it is too complicated for beginners.

The following basic programming topics are adopted for the summer program.

- Day 1: basic HTML tags, inputs, terminators, data types, variables, constants, operators, keywords, assignments, documents and writes.
- Day 2: functions, parameter passing, computation, return values, modularization, variable scope, good coding style.
- Day 3: logic conditions, if-else, nested if-else
- Day 4: loops and nested loops
- Day 5: one-dimensional array and index

As part of this learning, we will ensure students practice using good coding style (indents, meaningful variable names, comments) and proper prompt and meaningful output messages.

III. PROBLEM SOLVING AND DEBUGGING

Identifying the problems, developing the solutions, and troubleshooting are very important in learning any language in computer science. Students will be given several small programming assignments to solve for practicing each topic every day. Student should be able to run their programs and save all the source code examples at local computers. Here, some samples are provided for each topic. For each assignment after Day 1, students should call the function 3 times with different inputs and display their results.

Day 1 – HTML and variables

- Assignment 1.1: design a personal HTML page that displays the student's name in a large font size in red color as shown in Fig. 2.1.
- Assignment 1.2: design an HTML page that will use JavaScript to display the school name under the student's name in different color and size using `<font>` tags as shown in Fig. 2.2. When the school's name is clicked, the school should be launched in a new page.
- Assignment 1.3: design a week schedule listing the date at the top, hours in the left column, and activities in each cell in a HTML table format as shown in Fig. 2.3.
- Assignment 1.4: fix a given JavaScript program that has syntax errors – undefined variable, missing “;”, define a keyword as a variable.

Day 2 functions

- Assignment 2.1: define a function that takes a balance and annual interest rate, and returns the monthly interest as shown in Fig. 3.1.1. The source code example is shown in Fig. 3.1.2.
- Assignment 2.2: redo the assignment 2.1 by asking the user to enter the balance and interest rate in HTML `<INPUT>` and show the interest after user clicks on the button as shown in Fig. 3.2.1. The source code example is shown in Fig. 3.2.2.
- Assignment 2.3: redo the assignment 1 by using two sliders using `<INPUT type='range'>` to get the balance (range 500 – 5000 step 100) and interest rate (range 0.01 to 0.2 step 0.01) numbers. Define a new function and pass the two numbers to that function. Fig. 3.3.1 and 3.3.2 show the interface and source code example.

- Assignment 2.4: finish a given partial calculator in HTML. Test the program and verify the results by adding/subtracting several numbers.

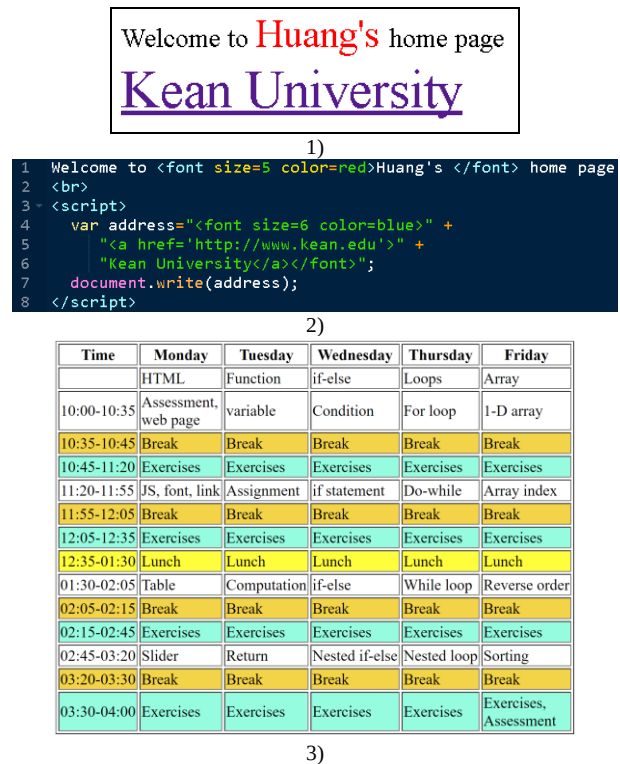


Fig. 2. 1). Assignment 1.1 and 1.2 – Creating an HTML web page; 2). The HTML source code for Fig. 2 1); 3). Assignment 1.3 - The activities are shown in an HTML TABLE format.

Balance: 1000
Annual interest rate: 0.03
Monthly interest: 2.5

Fig. 3.1.1. Assignment 2.1 - The balance and interest rates are hardcoded in a simple function.

```

1 <script>
2 var balance=1000;
3 var rate=3/100;
4 document.write("<br>Balance: "+balance);
5 document.write("<br>Annual interest rate: "+ rate);
6 document.write("<br>Monthly interest: " +
7   interest(1000,0.03));
8 - function interest(balance, rate) {
9   var interest=Number(balance)*Number(rate)/12;
10  return interest;
11 }
12 </script>

```

Fig. 3.1.2. The source code for Fig. 3.1.1.

Balance:
Annual interest rate:

Result:

Fig. 3.2.1. Assignment 2.2 - The balance and interest rate are entered through input boxes.

```

1 <script>
2 - function calInterest(balance,rate, box){
3   var result = Number(balance.value)*Number(rate.value)/12;
4   document.getElementById(box).value = result;
5 }
6 </script>
7 <form method="POST" name="myform">
8   Balance: <input type="text" size=5 name=balance>
9   <br>Annual interest rate:
10  <input type="text" size=5 name=rate>
11  <br><input type="button"
12    value="Click here to see the interest"
13    onClick="calInterest(balance, rate,'result');">
14  <br>
15  Result: <input type="text" size=6 name=result id=result>
16 </form>

```

Fig. 3.2.2. Assignment 2.3 - The source code for Fig. 3.2.1.

Fig. 3.3.1. Assignment 2.4 - The balance and interest rates are entered through sliders.

```

1 <script>
2 function calInterest(balance,rate, box){
3   var result = Number(balance.value)*Number(rate.value)/12;
4   document.getElementById(box).value = result;
5 }
6 function updateBalance(num) {
7   document.querySelector('#newBalance').value = num;
8 }
9 function updateRate(num) {
10  document.querySelector('#newRate').value = num;
11 }
12 </script>
13 <form name=myform2>
14 Balance:
15 <input type=range id=balance name=balance value=0.03
16 step=0.01 min=0.01 max=0.2 oninput="updateBalance(value)"
17 onChange="calInterest(balance, rate,'minterest');">
18 <output for=balance id="newBalance">0.03</output>
19 <br> Interest rate:
20 <input type="range" id="rate" name=rate value=1000
21 step=100 min=500 max=5000 oninput="updateRate(value)"
22 onChange="calInterest(balance, rate,'minterest');">
23 <output for=rate id="newRate">1000</output>
24 <br> Monthly interest:
25 <input type=text size=7 name=minterest id=minterest>
26 </form>

```

Fig. 3.3.2. The source code for Fig. 3.3.1.

Day 3 if-else

- Assignment 3.1: define a function that returns the bigger number of two given numbers. Ask user to enter two numbers using <INPUT>, pass the numbers to the function and display the bigger number as shown in Fig.s 4.1.1. Fig. 4.1.2 shows the source code.
- Assignment 3.2: redo assignment 3.1 to define a program that displays the biggest number of four given numbers. Fig.s 4.2.1 and 4.2.2 show the interface and source code examples.
- Assignment 3.3: define a function that takes student name, score and return the grade – A: 80 and above, B: 70 and above, C: 60 and above, F: below 60. Call the function 3 times with different input names and scores and display the inputs and their results. If the grade is F, display the output in red color for name, score and grade. Fig.s 4.3.1 and 4.3.2 show the interface and the source code examples.

Assignment 3.4: define a function that takes a person's name, blood pressure, weight (lb.) and height (inch), and returns a health condition – Good: healthy in blood pressure and healthy in BMI. Average: one of the BMI and blood pressure is healthy. Bad: both BMI and blood pressure are not healthy. Blood pressure healthy range is between 90 and 120 (include both ends). Numbers outside the range are not healthy. BMI = (weight in pounds x 703) ÷ (height in inches x height in inches). BMI healthy range is between 18.5 and 24.9. Any BMI outside that range is not healthy. Fig.s 4.1.1 and 4.1.2 show the interface and source code examples.

Fig. 4.1.1. Assignment 3.1 - Two numbers are entered in textboxes and the larger number is shown in a separate textbox.

```

1 <script>
2 function displayLarger(num1,num2,box) {
3   var larger=Number(num1.value);
4   if (Number(num2.value) > Number(num1.value))
5     larger= Number(num2.value);
6   document.getElementById(box).value = larger;
7 }
8 </script>
9 <form name=myform3>
10 Number 1: <INPUT type='text' name=num1 size=5>
11 <br>
12 Number 2: <INPUT type='text' name=num2 size=5>
13 <br>
14 <input type="button" value="Click here to see the larger number"
15 onClick="displayLarger(num1, num2,'larger');">
16 <br> The larger number is:
17 <input type=text size=5 name=larger id=larger>
18 </form>

```

Fig. 4.1.2. The source code for Fig. 4.1.1.

Fig. 4.2.1. Assignment 3.2 - Find and display the largest number from 4 input numbers.

```

1 <script>
2 function findLarger(num1,num2) {
3   var larger=num1;
4   if (Number(num2.value) > Number(num1.value))
5     larger= num2;
6   return larger;
7 }
8 function displayLarger(num1,num2,num3,num4,box) {
9   var larger=findLarger(findLarger(num1,num2), findLarger(num3,num4));
10  document.getElementById(box).value = Number(larger.value);
11 }
12 </script>
13 <form name=myform4>
14 Enter 4 numbers: <INPUT type=text name=num1 size=3>,
15 <INPUT type='text' name=num2 size=3>,
16 <INPUT type='text' name=num3 size=3>,
17 <INPUT type='text' name=num4 size=3>
18 <br>
19 <input type="button" value="Click here to see the largest number"
20 onClick="displayLarger(num1, num2, num3, num4,'larger');">
21 <br>The largest number is:
22 <input type=text size=5 name=larger id=larger>
23 </form>

```

Fig. 4.2.2. The source code for Fig. 4.2.1.

Fig. 4.3.1. Assignment 3.3 - Convert an input number into a letter grade.

```

1 <br><br>
2 <script>
3 function displayGrade(num,box) {
4   var grade="F";
5   var score=Number(num.value);
6   if (score>=80) grade="A";
7   else if (score>=70) grade="B";
8   else if (score>=60) grade="C";
9   else grade="F";
10  document.getElementById(box).value = grade;
11 }
12 </script>
13 <form name=myform5>
14 Enter the score: <INPUT type=text name=num size=3>
15 <br> <input type="button" value="Click here to see the grade"
16 onClick="displayGrade(num,'grade');">
17 <br>The grade is: <input type=text size=3 name=grade id=grade>
18 </form>

```

Fig. 4.3.2. The source code for Fig. 4.3.1.

Fig. 4.4.1. Assignment 3.4 - Determine the person's health condition based on 3 input numbers.

```

1- <script>
2- function displayHealth(blood,h,w,bmi_box,msg_box) {
3-   var message="Bad";
4-   var good_blood=false;
5-   var good_BMI=false;
6-   var BMI=(Number(w.value)*703)/
7-     (Number(h.value)*Number(h.value));
8-   if (Number(blood.value)>=90 && Number(blood.value)<=120)
9-     good_blood=true;
10-   if (Number(BMI)>=18.5 && Number(BMI)<=24.9)
11-     good_BMI=true;
12-   if (good_blood && good_BMI ) message="Good";
13-   else if (good_blood || good_BMI) message="Average";
14-   else message="Bad";
15-   document.getElementById(bmi_box).value = BMI;
16-   document.getElementById(msg_box).value = message;
17- }
18- </script>
19- <form name=myform6>
20-   Blood pressure: <INPUT type='text' name=blood size=3 >
21-   <br> Height (inch): <INPUT type='text' name=h size=3 >
22-   <br> Weight (lb): <INPUT type='text' name=weight size=3 >
23-   <br> <input type=button value="Click here to see the condition"
24-     onClick="displayHealth(blood,h,weight,'bmi','msg');" />
25-   <br> BMI is: <input type=text size=8 name=bmi id=bmi>
26-   <br> The health condition is:
27-   <input type=text size=3 name=msg id=msg>
28- </form>

```

Fig. 4.4.2. The source code for Fig. 4.4.1.

Day 4 loops

- Assignment 4.1: define a function that will return the sum of 1 to N – a given input number. Figs 5.1.1 and 5.1.2 show the interface and source code example.
- Assignment 4.2: define a function that will return total number of even number between two given numbers. Figs 5.2.1 and 5.2.2 show the interface and source code example.
- Assignment 4.3: write a program to print a half pyramid (triangle) in numbers with size a given input number N. Figs 5.3.1 and 5.3.2 show the interface and source code example.
- Assignment 4.4: define a function that will display the times table for numbers 1 to N – a given input number. Figs 5.4.1 and 5.4.2 show the interface and source code example.

Please enter a number N:

The sum of 1 to 10 is:

Fig. 5.1.1. Assignment 4.1 - Calculate and display the sum of the numbers 1 consecutively up to a given number N.

```

1- <script>
2- function displaySum(num,box) {
3-   var total=0;
4-   for (i=1;i<=Number(num.value);i++)
5-     total=total+i;
6-   document.getElementById("NN").innerHTML = num.value;
7-   document.getElementById(box).value = total;
8- }
9- </script>
10- <form name=myform7>
11-   Please enter a number N:
12-   <INPUT type='text' name=num size=3 >
13-   <br> <input type=button value="Click here to see the sum"
14-     onClick="displaySum(num,'sum');" />
15-   <br> The sum of 1 to <label id="NN">N</label> is:
16-   <input type=text size=4 name=sum id=sum>
17- </form>

```

Fig. 5.1.2. The source code for Fig. 5.1.1.

The begin number N1:

The end number N2:

The sum of the even numbers from 1 to 5 is:

Fig. 5.2.1. Assignment 4.2 - Calculate and display the sum of even numbers from N1 to the given number N2.

```

1- <script>
2- function displaySum(num1,num2,box) {
3-   var total=0;
4-   var i=Number(num1.value);
5-   do {
6-     if (i%2 ==0) total=total+i;
7-     i++;
8-   } while (i<=Number(num2.value));
9-   document.getElementById("N1").innerHTML = num1.value;
10-  document.getElementById("N2").innerHTML = num2.value;
11-  document.getElementById(box).value = total;
12- }
13- </script>
14- <form name=myform7>
15-   The begin number N1:
16-   <INPUT type='text' name=num1 size=3 >
17-   <br> The end number N2:
18-   <INPUT type='text' name=num2 size=3 >
19-   <br> <input type=button value="Click here to see the sum"
20-     onClick="displaySum(num1,num2,'sum');" />
21-   <br> The sum of the even numbers from
22-   <label id="N1">N1</label> to
23-   <label id="N2">N2</label> is:
24-   <input type=text size=4 name=sum id=sum>
25- </form>

```

Fig. 5.2.2. The source code for Fig. 5.2.1.

The size of triangle:

1
12
123
1234

Fig. 5.3.1. Assignment 4.3 - Print a triangle of a given size.

```

1- <script>
2- function displayTriangle(num) {
3-   var result="<br>\n";
4-   var i=1;
5-   while (i<=Number(num.value)) {
6-     for (j=1;j<=i;j++)
7-       result=result+j;
8-     result=result + "<br>\n";
9-     document.getElementById("result").innerHTML = result;
10-    i++;
11-  }
12- }
13- </script>
14- <form name=myform7>
15-   The size of triangle:
16-   <INPUT type='text' name=num size=3 >
17-   <br> <input type=button value="Click here to see the triangle"
18-     onClick="displayTriangle(num);" />
19-   <label id="result"></label>
20- </form>

```

Fig. 5.3.2. The source code for Fig. 5.3.1.

The size of times table:

	1	2	3	4	5	6	7	8	9
1	1x1=1	1x2=2	1x3=3	1x4=4	1x5=5	1x6=6	1x7=7	1x8=8	1x9=9
2	2x1=2	2x2=4	2x3=6	2x4=8	2x5=10	2x6=12	2x7=14	2x8=16	2x9=18
3	3x1=3	3x2=6	3x3=9	3x4=12	3x5=15	3x6=18	3x7=21	3x8=24	3x9=27
4	4x1=4	4x2=8	4x3=12	4x4=16	4x5=20	4x6=24	4x7=28	4x8=32	4x9=36
5	5x1=5	5x2=10	5x3=15	5x4=20	5x5=25	5x6=30	5x7=35	5x8=40	5x9=45
6	6x1=6	6x2=12	6x3=18	6x4=24	6x5=30	6x6=36	6x7=42	6x8=48	6x9=54
7	7x1=7	7x2=14	7x3=21	7x4=28	7x5=35	7x6=42	7x7=49	7x8=56	7x9=63
8	8x1=8	8x2=16	8x3=24	8x4=32	8x5=40	8x6=48	8x7=56	8x8=64	8x9=72
9	9x1=9	9x2=18	9x3=27	9x4=36	9x5=45	9x6=54	9x7=63	9x8=72	9x9=81

Fig. 5.4.1. Assignment 4.4 - Print a times table of a given size.

```

1- <script>
2- function displayTimesTable(num) {
3-   var result="<table border=1>\n";
4-   result=result+"<TR><TH>";
5-   for (i=1;i<=Number(num.value);i++)
6-     result=result + "<TH>" + i ;
7-
8-   for (i=1;i<=Number(num.value);i++) {
9-     result=result + "<TR><TH>" + i;
10-    for (j=1;j<=Number(num.value);j++)
11-      result=result+ "<TD>" + i + "x" + j + "=" + i*j;
12-    }
13-    result=result + "<TR>";
14-  }
15-  document.getElementById("result").innerHTML = result;
16- }
17- </script>
18- <form name=myform7>
19-   The size of times table:
20-   <INPUT type='text' name=num size=3 >
21-   <br> <input type=button value="Click here to see the times table"
22-     onClick="displayTimesTable(num);" />
23-   <label id="result"></label>
24- </form>

```

Fig. 5.4.2. The source code for Fig. 5.4.1.

Day 5 arrays

- Assignment 5.1: define a function that will return the smallest number in an input array string separated by ",". Split the array string into numbers and display the result. Figs 6.1.1 and 6.1.2 show the interface and source code example.
- Assignment 5.2: define a function that will find the given number in a given array string and return the index of the first match number. If the given number is not in the given array, display "Number xx was not found in this list" message. Figs 6.2.1 and 6.2.2 show the interface and source code example.
- Assignment 5.3: define a function that will reverse the order of a given array string separated by ",". Figs 6.3.1 and 6.3.2 show the interface and source code example.
- Assignment 5.4: define a function that will sort the numbers from low to high in a given array string separated by ",". Figs 6.4.1 and 6.4.2 show the interface and source code example.

Enter several numbers separated by ",": 3,16,4,9,1,8

Click here to find the smallest one

The smallest number is: 1

Fig. 6.1.1. Assignment 5.1 - Split the string into numbers and display the smallest number.

```

1 <script>
2 function displaySmallest(num) {
3   token=num.value.split(',');
4   var smallest=Number(token[0]);
5   for (i=0;i<token.length;i++)
6     if (Number(token[i])<smallest)
7       smallest=Number(token[i]);
8 }
9 document.getElementById("result").innerHTML = smallest;
10 }
11 </script>
12 <form name=myform8>
13   Enter several numbers separated by ",":
14   <input type='text' name=num size=10 >
15   <br> <input type=button value="Click here to find the smallest one"
16     onClick="displaySmallest(num);" >
17   <br>The smallest number is: <label id="result"></label>
18 </form>

```

Fig. 6.1.2. The source code for Fig. 6.1.1.

Enter several numbers separated by ",": 3,2,1,6,7

Enter the number for searching: 6

Click here to find the number

Number 6 was found at position: 4

Fig. 6.2.1. Assignment 5.2 - Split the string into numbers and display the position of the number for searching.

```

1 <script>
2 function displayPosition(num,knum) {
3   var message="Error";
4   var position=-1;
5   token=num.value.split(',');
6   for (i=0;i<token.length;i++)
7     if (Number(token[i])==Number(knum.value))
8       position=i;
9   if (position==1)
10    message="Number " + knum.value + " was not found in the list.";
11   else
12    message="Number " + knum.value + " was found at position: " +
13      (position+1);
14   document.getElementById("result").innerHTML = message;
15 }
16 </script>
17 <form name=myform9>
18   Enter several numbers separated by ",":
19   <input type='text' name=num size=10 >
20   <br>Enter the number for searching:
21   <input type='text' name=knum size=3 >
22   <br> <input type=button value="Click here to find the number"
23     onClick="displayPosition(num,knum);" >
24   <br><label id="result"></label>
25 </form>

```

Fig. 6.2.2. The source code for Fig. 6.2.1.

Enter several data separated by ",": NY,NJ,CA,FL,VA,TX

Click here to reverse the order

The reversed output is: TX,VA,FL,CA,NJ,NY

Fig. 6.3.1. Assignment 5.3 - Split the string into tokens and display the tokens in reversed order.

```

1 <script>
2 function displayReverse(arr) {
3   var narr=new Array;
4   token=arr.value.split(',');
5   for (i=0;i<token.length;i++)
6     narr[token.length-i-1]=token[i];
7   document.getElementById("result").innerHTML = narr;
8 }
9 </script>
10 <form name=myform10>
11   Enter several data separated by ",":
12   <input type='text' name=indata size=15 >
13   <br> <input type=button value="Click here to reverse the order"
14     onClick="displayReverse(indata);" >
15   <br>The reversed output is: <label id="result"></label>
16 </form>

```

Fig. 6.3.2. The source code for Fig. 6.5.

Enter several numbers separated by ",": 3,2,12,15,11,1,119

Click here to sort the numbers

The sorted output is: 1,2,3,11,12,15,119

Fig. 6.4.1. Assignment 5.4 - Split the string into numbers and display the numbers in order.

```

1 <script>
2 function displaySorting(arr) {
3   var tmp=0;
4   var token=new Array;
5   token=arr.value.split(',');
6   for (i=0;i<token.length-1;i++)
7     for (j=i+1;j<token.length;j++)
8       if (Number(token[i]) > Number(token[j])) {
9         tmp=token[i];
10        token[i]=token[j];
11        token[j]=tmp;
12      }
13 }
14 document.getElementById("result").innerHTML = token;
15 }
16 </script>
17 <form name=myform10>
18   Enter several numbers separated by ",":
19   <input type='text' name=indata size=15 >
20   <br> <input type=button value="Click here to sort the numbers"
21     onClick="displaySorting(indata);" >
22   <br>The sorted output is: <label id="result"></label>
23 </form>

```

Fig. 6.4.2. The source code for Fig. 6.4.1.

IV. PROGRAM DESIGN

In order to make the program suitable for high school students, the schedule is designed so that neither lectures nor exercises take too long. There will be no homework for students to take home, because they might not have computers or internet access at home.

A. Program Schedule

This paper proposes the summer program to consist of 25 hours over 5 days. Each day will include 5 hours of lecture and lab exercises. To ensure students are able to fully understand the concepts, we recommend 30 minutes of assignment exercises for every 35 minutes of lecture, with 10-minute breaks. That means there will be 4 periods with 75 minutes per period each day. Students should finish the lab assignments at school. Since it is possible that students might need to take public transportation, the operation hours are proposed from 10 am to 4 pm, including a 1-hour lunch break. If the program doesn't have funding for providing free lunch, students should bring their own lunch.

B. Classroom Equipment

Since students are not expected to bring laptops, the classroom should have computers connected to the internet. The computers only need a browser that can run HTML and JavaScript. No other software is required. The operating system can be Windows, Linux, or iOS. There should be a projector and screen for the instructor to present slides and demonstrate the code, examples and results.

C. Teaching Assistant

In addition to the instructors, several computer science major college students should be recruited as the teaching assistant (TA) to help students for lab exercises. The TAs should know HTML and JavaScript, and they should review the curriculum, lab assignments, and instructor's teaching materials one day before the class. To ensure the learning outcomes are achieved, we recommend a TA/student ratio of 1:5.

V. PROGRAM ASSESSMENT

We propose a short assessment when the program starts during the first period of the first day. Students will answer 10 questions related to computing topics, such as variables, assignments, functions, etc. There will be another assessment on the last period of the last day. Students will answer 20 questions that include 10 questions similar to the first assessment, and 10 new questions. These two assessments will help to evaluate the effectiveness of the summer program and the analytical results can help to improve the future program design and curriculum.

VI. CONCLUSIONS

We propose a Computer Science summer program that will enable participating students to learn the foundations of computer programming in 5 days. This will provide them valuable exposure to computing in a short period of time and better prepare them to pursue studies or a career in software development—which can be a pathway for disadvantaged students to quickly move to the middle class. Universities have facilities and resources—because most offer a Computer Science major—so they are in a better position to assist these students. This paper proposes using HTML and JavaScript as the first computer language for students to learn computer programming. Though these are introductory languages, we believe it is more important to trigger students' interests in

coding than to try to push them with a complicated language. Five days of curriculum and assignments are proposed in this paper such that students can use the Online JavaScript Editor to write programs and see the results immediately. Next steps from this paper will be to use the proposed curriculum to run the program—and evaluate the results to understand whether learning outcomes were achieved.

REFERENCES

- [1] Code.org
- [2] J. Swartz, "Businesses say they just can't find the right tech workers," *USA TODAY*, March 28, 2017.
- [3] College Board, "AP program participation and performance data 2017," Program Summary Report, 2017.
- [4] X.-L. Chen and M. Soldner, "STEM attrition: College students' paths into and out of STEM fields," Statistical Analysis Report, National Center for Education Statistics, 2013.
- [5] C.-Y. Huang, J.-. Liou, and G. Huang, "Pathway from learning multimedia software to computer science education," in *Proc. the 11th International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS'15)*, Las Vegas, Nevada, July 27-30, 2015.
- [6] C.-Y. Huang, R. Olszewski, and R. Tomazic, "An integrated multimedia computer science summer program for middle school students," in *Proc. the 12th International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS'16)*, Las Vegas, July 25-28, 2016.
- [7] HTML color. [Online]. Available: <http://htmlcolorcodes.com/>
- [8] Online Javascript Editor. [Online]. Available: https://www.tutorialspoint.com/online_javascript_editor.php



Ching-yu Huang is an assistant professor of the School of Computer Science at Kean University, Union, New Jersey, USA since September 2014. Dr. Huang received a Ph.D. in computer & information science from New Jersey Institute of Technology, Newark, New Jersey, USA.

Prior to joining Kean University, Dr. Huang had more than 16 years of experience in the industry and academics in software development and R&D in bioinformatics. His research focuses SNP genotype calling and cluster detection; image processing and pattern recognition, especially in microarray and fingerprint; geotagged images and location information reconstruction; database application development; data processing automation; E-learning, educational multimedia, methodology, and online tools for secondary schools and colleges. Dr. Huang has more than 40 publications in journals and conferences and more than 20 presentations in workshops and invited lectures.



Mayra Bachrach is a lecturer of the School of Computer Science at Kean University, Union, New Jersey, USA. Professor Bachrach is the PI for a 2018 Google CS Educator grant to provide professional development to middle school teachers. Prior to joining the Kean Faculty, she taught computer science and robotics for grades 8-12.